Function And Procedure

Tuesday, August 24, 2021 6:14 PM

Structured Programming

Concept Explanation

$F(x) = x^2 + 2x + 3x + 4x + 5$

Basically I gave name to some instruction and can use it to call again again

F(1)

↳ Parameter

F(2)

Function

Procedure

Returns a value

does not return a value

Header Of Function.

Parameters acts as input.

variable that will be the Answer

Function Name (Name of variable : datatype , Name of variable : Datatype) Return Datatype

End Function.

$F(x) = x^2$ $F(2) = 2^2 = 4$

Question : Write a function that take x as parameter and return its square.

```
Function Square (ByVal x : integer) Return integer.
    sq = x * x
    Return sq
EndFunction.

Ans = square(5)
Print Ans
```

= 25

Note : Same header and footer will be use for procedure.

```
Procedure Name (Name of variable : datatype , Name of variable : Datatype)

EndProcedure.
```

Functions And Procedure With Arrays

6 A company hires out rowing boats on a lake. The company has ten boats, numbered from 1 to 10.

The company is developing a program to help manage and record the hiring out process.

Hire information is stored in three global 1D arrays when a boat is hired out. Each array contains 10 elements representing each of the ten boats.

The three 1D arrays are summarised as follows:

Array	Data type	Description	Example data value
HireTime	STRING	The time the boat was hired out	"10:15"

6 A company hires out rowing boats on a lake. The company has ten boats, numbered from 1 to 10.

The company is developing a program to help manage and record the hiring out process.

Hire information is stored in three global 1D arrays when a boat is hired out. Each array contains 10 elements representing each of the ten boats.

The three 1D arrays are summarised as follows:

Array	Data type	Description	Example data value
`HireTime`	`STRING`	The time the boat was hired out	`"10:15"`
`Duration`	`INTEGER`	The number of minutes of the hire	`30`
`Cost`	`REAL`	The cost of the hire	`5.75`

If an individual boat is not currently on hire, the corresponding element of the `HireTime` array will be set to `"Available"`.

The programmer has started to define program modules as follows:

Module	Description

`AddTime()`	• Called with two parameters:
	  ○ a `STRING` value representing a time
	  ○ an `INTEGER` value representing a duration in minutes
	• Adds the duration to the time to give a new time
	• Returns the new time as a `STRING`

`ListAvailable()`	• Called with a `STRING` value representing the time the hire will start
	• Outputs the boat numbers that will be available for hire at the given start time. A boat will be available for hire if it is either:
	  ○ currently not on hire, or
	  ○ due back before the given hire start time
	• Outputs the number of each boat available
	• Outputs the total number of boats available or a suitable message if there are none

`RecordHire()`	• Called with four parameters:
	  ○ an `INTEGER` value representing the boat number
	  ○ a `STRING` value representing the hire start time
	  ○ an `INTEGER` value representing the hire duration in minutes
	  ○ a `REAL` value representing the cost of hire
	• Updates the appropriate element in each array
	• Adds the cost of hire to the global variable `DailyTakings`
	• Converts the four input parameters to strings, concatenated using commas as separators, and writes the resulting string to the end of the existing text file `HireLog.txt`

(a) Write **pseudocode** for the module `ListAvailable()`. (8)

```
PROCEDURE ListAvailable ( starttime :INTEGER)

BoatNumAvailable = 0
FOR index = 1 TO 10
    Available = False
```

```
Available = False
IF Hiretime[Index] = "Available"
    Then
        Available = True
    Else
        Timeback = Addtime(Hiretime[Index], duration[index])
        IF Timeback < starttime
            Then
                Available = True
        ENDIF
    ENDIF
    IF Available = True
        Then
            Output "Boat ", Index, " IsAvailable"
            BoatNumAvailable = BoatNumAvailable + 1
    EndIF
ENDFOR
IF BoatNumAvailable > 0
    Then
        Outp "There are", BoatNumAvailable, " boats available"
    Else
        output "Sorry, No boats"
    ENDIF
ENDPROCEDURE
```

FUNCTIONS AND PROCEDURE WITH FILES

6 Account information for users of a library is held in one of two text files; `UserListAtoM.txt` and `UserListNtoZ.txt`

The format of the data held in the two files is identical. Each line of the file is stored as a string that contains an account number, name and telephone number separated by the asterisk character (`'*'`) as follows:

```
<Account Number>'*'<Name>'*'<Telephone Number>
```

An example of one line from the file is:

```
"GB1234*Kevin Mapunga*07789123456"
```

The account number string may be **six** or **nine** characters in length and is **unique for each person**. It is made up of alphabetic and numeric characters only.

An error has occurred and the same account number has been given to different users in the two files. There is **no** duplication of account numbers **within each individual file**.

A program is to be written to search the two files and to identify duplicate entries. The account number of any duplicate found is to be written to an array, `Duplicates`, which is a 1D array of 100 elements of data type `STRING`.

The program is to be implemented as several modules. The outline description of three of these is as follows:

Module	Outline description
`ClearArray()`	• Initialise the global array `Duplicates`. Set all elements to the empty string.

`FindDuplicates()`	• Read each line from the file `UserListAtoM.txt`
	  ◦ Check whether the account number appears in file `UserListNtoZ.txt` using `SearchFileNtoZ()`
	  ◦ If the account number does appear then add the account number to the array.
	• Output an error message and exit the module if there are more duplicates than can be written to the array.

`SearchFileNtoZ()`	• Search for a given account number in file `UserListNtoZ.txt` ◦ If found, return `TRUE`, otherwise return `FALSE`

(a) State **one** reason for storing data in a file rather than in an array.

.....

..... [1]

© UCLES 2019

(b) Write **program code** for the module `SearchFileNtoZ()`.

Visual Basic and Pascal: You should include the declaration statements for variables.
Python: You should show a comment statement for each variable used with its data type.

Programming language .....

Program code

.....

.....

.....

.....

.....

.....

[7]

(c) Write **pseudocode** for the module `FindDuplicates()`.

The module description is given in the table on page 12.

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

..... [8]

(d) `ClearArray()` is to be modified to make it general purpose. It will be used to initialise any 1D array of data type `STRING` to any value.

It will now be called with three parameters as follows:

1. The array
2. The number of elements
3. The initialisation string

You should assume that the lower bound is 1.

(i) Write **pseudocode** for the modified `ClearArray()` procedure.

.....

.....

.....

.....

.....

.....

.....

..... [3]

(ii) Write **program code** for a statement that calls the modified `ClearArray()` procedure to clear the array `Duplicates` to `"Empty"`.

Programming language .....

Program code

.....

..... [2]

Appendix

Built-in functions (pseudocode)

Each function returns an error if the function call is not properly formed

Appendix

Built-in functions (pseudocode)

Each function returns an error if the function call is not properly formed.

`MID(ThisString : STRING, x : INTEGER, y : INTEGER) RETURNS STRING`
returns a string of length `y` starting at position `x` from `ThisString`

Example: `MID("ABCDEFGH", 2, 3)` returns `"BCD"`

`LENGTH(ThisString : STRING) RETURNS INTEGER`
returns the integer value representing the length of `ThisString`

Example: `LENGTH("Happy Days")` returns `10`

`LEFT(ThisString : STRING, x : INTEGER) RETURNS STRING`
returns leftmost `x` characters from `ThisString`

Example: `LEFT("ABCDEFGH", 3)` returns `"ABC"`

`RIGHT(ThisString: STRING, x : INTEGER) RETURNS STRING`
returns rightmost `x` characters from `ThisString`

Example: `RIGHT("ABCDEFGH", 3)` returns `"FGH"`

`INT(x : REAL) RETURNS INTEGER`
returns the integer part of `x`

Example: `INT(27.5415)` returns `27`

`NUM_TO_STRING(x : REAL) RETURNS STRING`
returns a string representation of a numeric value.
Note: This function will also work if `x` is of type `INTEGER`

Example: `NUM_TO_STRING(87.5)` returns `"87.5"`

`STRING_TO_NUM(x : STRING) RETURNS REAL`
returns a numeric representation of a string.
Note: This function will also work if `x` is of type `CHAR`

Example: `STRING_TO_NUM("23.45")` returns `23.45`

`ASC(ThisChar : CHAR) RETURNS INTEGER`
returns the ASCII value of `ThisChar`

Example: `ASC('A')` returns `65`

`CHR(x : INTEGER) RETURNS CHAR`
returns the character whose ASCII value is `x`

Example: `CHR(87)` returns `'W'`

© UCLES 2019

`UCASE(ThisChar : CHAR) RETURNS CHAR`
returns the character value representing the upper case equivalent of `ThisChar`
If `ThisChar` is not a lower case alphabetic character, it is returned unchanged.

Example: `UCASE('a')` returns `'A'`

Operators (pseudocode)

Operator	Description
`&`	Concatenates (joins) two strings Example: `"Summer" & " " & "Pudding"` produces `"Summer Pudding"`
`AND`	Performs a logical `AND` on two Boolean values Example: `TRUE AND FALSE` produces `FALSE`
`OR`	Performs a logical `OR` on two Boolean values Example: `TRUE OR FALSE` produces `TRUE`

© UCLES 2019